INF367 - Selected Topics in AI

Learning Theory and Neuro-symbolic AI

Assignment 2

September 8, 2020

This assignment contributes to 6% of the final grade. Your grade will be based on the clarity and correctness of the results. You may talk with your colleagues but you should understand yourself all the steps of your answers/code and write with your own words. Copying is not allowed.

Deliverables [1]:

- A jupyter notebook containing all the code to reproduce your work and
- a report explaining your answers/code.

1 PAC Learning (Points 1)

1. Let $F = (E, H, m)$ be the learning framework where
    - H is the set of all conjunctions of literals that can be formulated in $V = v_1, \ldots, v_n$;
    - E is the set of all valuations in V;
    - m is a function that maps each conjunction of literals in H to the valuations that satisfy it.

    Describe *with your own words* the polynomial time algorithm in the slides of Lecture 7 for PAC learning F with classification noise. Point to the relevant literature showing correctness.

2. Give an upper bound for the VC dimension of F and explain why your upper bound is correct (see Slide 23 of Lecture 6).

[1]Please contact the lecturer if you would like to deliver your assignment in another format (e.g., another programming language).

2 Hands on Code (Points 5)

Implement an algorithm for PAC learning the learning framework presented in Section 1 (conjunctions of literals) and an "oracle" that returns examples with noisy classification according to a probability distribution. Your program should also be able to estimate the error of the hypothesis.

Input: The parameters ϵ (upper bound for the error of the hypothesis), δ (where $1 - \delta$ is the confidence of the hypothesis), η_b (upper bound for the classification noise rate), the size k of the training set. The parameters to define the probability distribution (if needed).

Guidelines: These are guidelines for the implementation.

- How to select the target:

  You can generate the target randomly or fix one in your code (suggestion: use $n = 10$ variables).

- How to generate the examples randomly:

  You can choose any distribution to generate your sample, as long as the distribution used for training is the same as the one used for estimating the error of the hypothesis.

- How to select the hypothesis:

  Generate a set of examples of size k (given as input). Use the target to label the examples and misclassify the label with chance η_b. Create a hypothesis by applying the strategy in the slides of Lec 7 (or see Section 3 of "Learning from Noisy Examples" by Angluin et al.).

- How to estimate the error of the hypothesis:

  The error of a hypothesis h w.r.t. a target t and a probability distribution D—in symbols, $D(m(t) \oplus m(t))$—can be difficult to compute. You can estimate it as follows. Pick a set S of examples of size N generated according to D. Calculate $\hat{t} = \frac{|S \cap (m(t) \oplus m(h))|}{N}$. It can be shown that $\hat{t}$ is an unbiased estimator of $D(m(t) \oplus m(h))$. Choose $N = 40 * k$, where k is the size of the training set. Alternatively, if you are using the uniform distribution and n variables, your program can calculate $D(m(t) \oplus m(h))$ by iterating over all 2^n examples.

Output: The hypothesis and an estimate of the error of the hypothesis w.r.t. the target and the probability distribution used for generating the training set.

Generalization guarantee: Fix $\epsilon = 0.3$ and $\delta = 0.1$ and try to find η_b and an estimate the size k of the training set such that with probability at least $1 - \delta$ the error of the hypothesis should be bounded by ϵ. Your program should try to estimate k by running the algorithm $10/\delta$ times and showing that in at most 10 out of $10/\delta$ times the error of the hypothesis is greater than ϵ. Choose 3 values for k to perform your experiments.